DOCUMENT RESUME

ED 102 969 IR 001 651

AUTHOR	Nutt, Gary J.
TITLE	Computer System Resource Requirements of Novice Programming Students.
INSTITUTION	Colorado Univ., Boulder. Dept. of Computer Science.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	CU-CS-039-74
PUB DATE	Feb 74
NOTE	32p.
EDRS PRICE	MF-$0.76 HC-$1.95 PLUS POSTAGE
DESCRIPTORS	College Students; *Computer Programs; Computers; *Computer Science Education; Data Processing; Higher Education; Information Scientists; Information Systems; Input Output; Programers; *Programing; Resource Centers
IDENTIFIERS	FORTRAN; *Minicomputers

ABSTRACT

The characteristics of jobs that constitute the mix for lower division FORTRAN classes in a university were investigated. Samples of these programs were also benchmarked on a larger central site computer and two minicomputer systems. It was concluded that a carefully chosen minicomputer system could offer service at least the equivalent of the central system, and that certain aspects could be distinctly better. (Author/SK)

ED102969

BEST COPY AVAILABLE

Computer System Resource Requirements of Novice Programming Students *

by

Gary J. Nutt
Department of Computer Science
University of Colorado
Boulder, Colorado 80302

Report #CU-CS-039-74 February, 1974

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR 001 651

* This work was funded by National Science Foundation Grant #GJ-660

ABSTRACT

The characteristics of jobs that constitute the mix for lower division FORTRAN classes in a university are investigated. Samples of these programs are also benchmarked on a large central site computer and two minicomputer systems. The conclusion of this study is that a carefully chosen minicomputer system could offer service at least the equivalent of the service provided by the central site system, and that certain aspects of this service could be distinctly better.

CR Categories: 1.5, 2.4, 6.2

In a time of constant (or decreasing) enrollment in many universities, one might not expect enrollment for a given set of courses to increase significantly. The University of Colorado has been operating at an imposed enrollment of 20,000 students since the Fall Semester of 1972; nevertheless, all of the lower division computer classes have been subject to higher enrollments in corresponding academic terms from year-to-year. In particular, the lower division FORTRAN class enrollment for the 1971-1972 academic year was 765 students; for the 1972-1973 academic year, 874 students; and the 1973-1974 academic year enrollment figure is 950 students.* Each of these students write and test an average of twelve programs per year, with each program requiring about 5 debugging runs. This large (and growing) novice student job load requires a significant amount of computer center resource in terms of operator time, line printer and card reader use, keypunch, etc. It is our contention, however, that an insignificant amount of processor time is required to support this mix.

A few critical characteristics of lower division FORTRAN programs will be explored in some depth later in this paper; we now only briefly define the

---

* The Department of Computer Science lower division FORTRAN courses are:

Computer Science 162 - Digital Computer Programming
Computer Science 201 - Introduction to Computer Science for Scientists and Engineers
Computer Science 203 - Introduction to Computer Science for Liberal Arts Majors
From Fall, 1971 through Fall, 1972, Computer Science for Business Majors was also a FORTRAN course.

character of a typical program from this mix. The average job includes a call to the compiler, possibly followed by loading and execution. The total amount of central processor time, on the average, is less than one second; a portion of this processing requirement is absorbed in compilation and loading. The programming assignments become more complex late in the academic term, requiring more run time, (the average central processor time is still less than two seconds). The reasons for this light processor requirement are concerned with the assignments; iterative calculations are generally contrived and converge rapidly, or the error bound is very generous. (We should also point out that error bounds are never seriously considered in the lower division programming classes.) On the other hand, the average program from this mix is composed of about 30 cards, and produces about 140 lines of output. It is easy to recognize the imbalance between the time required for printing and the time required for processing.

At the University of Colorado, as in many universities, lower division programming courses use the central site computer, in this case a pair of Control Data 6400 systems. This particular installation supports both batch and interactive processing, although the introductory students use only the batch processing facility. In addition to the educational load, the system is subjected to a variety of other tasks; it must support research projects involving large numerical computations; it is likely that administrative data processing may be done on the machine, etc. This wide range of responsibility can only be covered at the expense of individual subsets of the set of all users. As the work load subjected to the system increases, turnaround time for individual jobs may be expected to increase. Features of the system that

might be oriented toward the novice programmer may be sacrificed for the good of the community. For example, optimizing compilers may replace those that produce object code more rapidly; run time error messages may be terse and uninterpretable; the job control language may be so complex as to make a simple compile-and-execute sequence a formidable task; the job accounting algorithms may be unrealistic.

One alternative to central site computing is the use of a dedicated minicomputer system for lower division programming students. This situation can improve the cost-effectiveness of computation by using relatively inexpensive processing power to achieve a closer match between job input/output time and the processor utilization. However the use of minicomputer systems may actually create a situation in which the bottleneck for throughput is the processor, i.e., compilation, loading, and execution of a job may actually require more time than reading the deck of cards or printing the resulting listing.

Nevertheless, we contend that this alternative can be superior to the use of a central site system if the minicomputer system is carefully chosen. In the remainder of this paper we shall support this contention by discussing a job mix analysis done on lower division FORTRAN programs executed on the Control Data 6400 batch system. We shall also discuss a more cost-effective method of offering the same service with the added benefit of minimal hands-on experience.

JOB LOAD ANALYSIS

A characterization of the lower division FORTRAN job load, for the purpose of this study, is primarily concerned with indicators of I/O time and processor utilization. It is desirable to obtain a quantification of the expected number of input cards for a program, the number of lines to be printed, and the amount of processor time spent in satisfying computational needs of the average program, as well as corresponding maximum loads. It might appear that we should also be interested in the mix of machine instructions used during program execution, so that comparisons between large system execution and minicomputer executions of the same programs could be made. Some simple benchmark programs illustrate that this is not necessary.

An interesting characteristic of a job run on the central site computer would be the expected actual turnaround time, although we have not been able to obtain accurate quantification of this measure of performance. We would also like to obtain true job costs, although this, too, is difficult to obtain, due to the somewhat artificial educational computer time rates that usually exist in this type of environment.

To achieve the goals for load characterization, we used two standard monitoring tools [2]. The system accounting log is exploited to obtain a reasonably conclusive set of data concerning I/O characterization, central processor times, accounting charges, and throughput requirements. Next homework assignments that constitute the mix were written and benchmarked on the Control Data 6400 system. The programs were then tested on two available minicomputer systems to compare execution times.

Accounting Log Analysis

All student jobs that are executed on the Control Data 6400 system at the University of Colorado must supply accounting information designating a "user number" and a "subaccount number". Each course is assigned a user number, and each student is assigned his own subaccount number. Thus the accounting system allows job charges to be made directly to the individual student subaccount. This accounting procedure proves to be useful in associating monitoring information with an individual, as will be discussed below.

Each job executed on the system produces a trace of messages for accounting purposes, as well as operator information and a list of control cards. This extended accounting log, or dayfile, has been shown to be quite useful for analyzing machine performance [1, 4]. However, the volume of information written to the dayfile precludes the possibility of saving any significant number of past records. The University of Colorado Computing Center has recognized this problem, and has taken measures to preserve certain items of information from the system dayfile for long periods of time [3]. The process is to perform an analysis of the system dayfile, gathering a set of characteristics for a job, condensing these characteristics to a 136 character string, and saving the resulting strings on a magnetic tape file called the Job Master File. Table 1 summarizes the pertinent information contained in a record on the Job Master File. The practice of producing this condensed file was initiated in January, 1972, and thus all information (shown in Table 1) for each job executed on the system since that time is available for analysis. It is this Job Master File that we have used to obtain information about the job load caused by FORTRAN programs.

The analysis program utilizes the user number field to recognize applicable records from the Job Master File. The subaccount field is used to distinguish between jobs submitted by the various students enrolled in any given class. The following illustrations of data correspond to the set of student programs for Computer Science 201, "Introduction to Computer Science for Scientists and Engineers" for the Spring Semester (approximately January 15 through May 15) of 1973. Data gathered for other courses and time periods is typefied by this particular sample.

In the following presentations, the histograms are for the job mix over the entire semester and we also present some histograms for the month of May, which represents the maximum demand placed on the system by the C.S. 201 students. The maximum demand month is analyzed since it is this load that will place the highest demand on the dedicated minicomputer (i.e., it is the "worst case" load).

During the Spring Semester of 1973 there were 18164 jobs submitted under the C.S. 201 user number. There were 287 students enrolled in the course approximately two weeks after the term began, hence each student submitted an average of 63.3 jobs for the semester. In the Job Master File analysis program, we compute the mean number of times that a subaccount number (i.e., an individual student account number) appeared in any given day. This measure was deemed worth obtaining, since our experience (and Teaching Assistants) told us that students often did their entire assignment in one day. The mean varied from 2.6 jobs/day for each student in January to 5.0 jobs/day for each student in May, with a mean over the entire semester of 3.6 jobs/day.

The batch system spools card images onto the disk for subsequent memory and processor assignment. Disk space is allocated to each job in 64 word (640 character) units. The Job Master File records this sector allocation rather than the actual number of cards read for each job, and it is that measure that is given here. Observations indicate that batch jobs require at least three sectors, i.e., there is a two sector overhead figure added into each job allocation.* Thus, for 80 column card input, the number of cards in i sectors is 8(i-3) + 4, where 4 is the expected number of cards in the last sector.

Figure 1 is a histogram of the number of input sectors allocated per job, e.g., about 13% of the 18164 jobs (2326) were allocated more than three sectors and less than or equal to 4 sectors. The arithmetic mean number of disk sectors allocated for card images was 184.8; the histogram indicates a median value of about 6 sectors/job. The explanation for this difference is that a small number of jobs, 65, were allocated more than 26 sectors, and have biased the mean value substantially. In our characterization of the typical job we shall use the median value; a sector count of 6 is equivalent to about 30 cards per job. This number of cards includes control cards, program, and data.

Figure 2 shows the statistics for disk sectors used to hold the input stream for jobs submitted during the month of May. Again we note an inflated mean value and a median value of about 6 sectors.

The output spooling process again uses 640 character disk sectors to hold line printer images, although a packed format is used for the representation of a line. Our investigation into the average number of characters per line

---

*19 jobs in the sample period were allocated less than or equal to 1 sector; this inconsistency is caused by interactive jobs on the student account number, probably entered by the instructor.

indicates that a line is about 50 characters long. This observation agrees with an independent study carried out by the Computing Center.* With this in mind, the expected number of lines for j sectors is

$$\frac{640 \text{ character/sector}}{50 \text{ character/line}} \text{ j sectors} = 12.8 \text{ j lines.}$$

Figure 3 indicates our finding over the entire semester, and Figure 4 gives the corresponding results for the month of May. The mean value for output sector allocation of 13.9, again, is biased by a few jobs with an inordinately high allocation of output sectors (33 jobs were allocated more than 150 sectors for output). However, the median value is close to the calculated arithmetic mean, compared to an expected number of lines per job of 178. The corresponding mean for May was 19 sectors, (243 lines) prompting Figure 5 which shows the mean number of output sectors allocated to each job for each month in the sample period. The conclusion is that the amount of output produced by a typical job is sensitive to the point in the semester; a similar analysis indicates that the number of cards is relatively constant over the same period of time.

A measure of the amount of time that the job competes for the processor is the amount of central processor time charged to the job. This measure ignores the amount of I/O, particularly in the Control Data 6400 with its peripheral processors. In Figure 6, the histogram for central processor time is given. Note that 88.5% of all jobs submitted during the semester required less than or equal to one second of central processor time. This processor charge includes compilation, loading, and execution. Almost 95% of the jobs

---

* This work was done by C. J. Brauch, Assistant Director, University of Colorado Computing Center.

require less than two seconds and almost all of the remaining jobs ran to time limit expiration, as might be expected in an introductory programming course. The mean central processor time for the entire semester was 1.0 seconds, while the mean central processor times for the individual months were 0.25 seconds, 0.60 seconds, 0.67 seconds, 0.93 seconds, and 1.7 seconds for the respective months of January through May. The histogram for processor time during the month of May does not differ substantially from Figure 6, although the mean has increased.

Data was gathered and analyzed to represent the amount of time that a job was resident either in an input/output queue, or in competition for central processor and disk. From practical observation of computer center operation, these results were rejected as meaningless for this study. The average amount of "queue-to-queue" time for our class of jobs was only 14 seconds; due to the process by which job decks are submitted, run, and output returned to the user, the "actual turnaround time" is closer to 15 minutes to a half hour. Thus we could not justify using these figures to reflect a turnaround figure that represented true service to the user.

The accounting algorithm used to compute dollar charges for a job uses the parameters: average amount of memory allocated, central processor time, mass storage transfers, card sectors allocated, and output sectors allocated. The exact coefficients for these parameters have varied with the three academic years we have mentioned, and with certain other factors such as remote job entry site. Nevertheless, we did monitor the job dollar charges to get an indicator of the way in which our educational computer time budget is spent. Figure 7 is a histogram of dollar charge per job, for the 18164 jobs. The

mean cost was about 34¢, for a total allocation of $6257.92 for the semester (this figure was obtained from billing invoices, and agrees with our monitored data). The resulting dollar cost per student for the term was $21.80 for computer time. Considering the combination of all lower division FORTRAN classes, the allocation amounted to $7190.45 or about $18.80 per student. Corresponding charges for the fall semester (under the same charging algorithm) were $6578.28 total, or $17.40 per student. Both the previous year (1971-72 academic year) and the following year saw higher charging rates. The total amount of money spent on the lower division FORTRAN computer time amounted to $13768.73 for the 1972-1973 academic year (9 months).

A final conclusion about the data gathered from the Job Master File is concerned with input-process-output overlap times. Using the mean values for processor time and output lines, and the median value for cards read, we can calculate card reader and line printer rates required to achieve a maximum overlap of utilization time. Assuming one second of processor time (utilized in about 2 seconds of real time) the card reader must read 30 cards and the line printer must print 180 lines in the corresponding real time period. Thus the card reader must operate at a constant 9000 cards per minute and the line printer (ignoring page eject time) must operate at a constant 5400 line per minute to keep up with the processor.

The Benchmark Tests

This portion of the study was necessary in order to obtain an understanding of processor requirements of the FORTRAN programs on some typical minicomputer systems. Four programming assignments were chosen from the lower division FORTRAN course work; the assignments varied from the initial assignment to the final assignment for the academic semester. These assignments were then programmed and tested on the Control Data 6400 batch system in order to compare these properties with the data obtained from the Job Master File analysis. The accounting log provides the amount of central processor time charged to the job for compilation and for the load and execute phase of the job. The time of day in seconds is also available from this log, thus we have a clumsy estimate of real time requirements. The amount of real time required for processing never exceeded two seconds for any single phase of job execution, even though the job was serviced in a multiprogramming environment.

The four programs were then tested on a "large minicomputer" system considered by some to be a "midi-computer"; we shall call this computer X. Average instruction execution times and memory cycle time are not relevant here (as will be seen below), since the majority of processor time is spent in the compiler and loader. This system also printed each line on the line printer after compilation of that statement; this tends to have a detrimental effect on the amount of time required to compile the code.

Finally, three of the four programs were tested on another classic minicomputer (which we shall call Y) which has gained a reputation for inefficient software. This system compiled the code with no listing until the compilation process was complete.

The four test programs are described below, and the benchmark findings are recorded in Table 2.

The first test program is essentially a keypunching assignment; it consists of eight cards containing FORTRAN statements and four job control cards. This program produced about 80 lines of output spread over four pages, including twenty lines on a banner page, two pages of listing, and an additional page with 20 lines of accounting information and control card information

The second test program would typically have been assigned in the second month of the course. The student is given a crude definition of a prime number, and he is asked to generate the primes that are less than or equal to twenty. This program was consciously written to look like code produced by a novice programmer, and contains three levels of nested loops. Our test version was composed of 18 FORTRAN cards and 4 control cards, and produced 95 lines of output on the central site machine.

Program number 3 would be assigned near the middle of the course, and is an output formatting exercise. The deck contains 30 FORTRAN statements, 4 control cards, and one data card; the output consists of about 160 lines including the 40 lines of banner and accounting information.

The last program is a simulation of a random walk on a 30 x 40 grid, with the marker initially placed in the center of the grid. The program was to execute until the marker exited successfully "through a door" or unsuccessfully "through a wall". The program that executed on the Control Data 6400 had 59 FORTRAN cards, 4 control cards, and one data card. There were about 170 lines of output. The code used a random number generator which was not immediately available on either of the minicomputers, thus there is no execution time on minicomputer X, and no test of the program on minicomputer Y.

The data in Table 2 indicates the compiler on minicomputer X is substantially faster than the compiler for minicomputer Y. It is also apparent that FORTRAN programs that are larger than, say, 100 cards should not be compiled repeatedly on these systems due to the excessive compile time. The load times are relatively constant for each minicomputer, i.e. X loads in about 12-13 seconds and Y loads in about 35 seconds. It is likely that disk transfer rates and library sizes determine these times. With the exception of the execution of TEST3 on minicomputer X, all of the execution times are insignificant when compared with compile and load times. We are unable to explain the inordinately large execution time for TEST3 on X.

A similar benchmark test was done on a third minicomputer system with a different set of programs.* The first program tested evaluated a 10 x 10 determinant with floating point numbers. Gary reports that the program consisted of 113 cards, required 7 seconds to compile (without listing), 15 seconds to load, and less than one second to execute. He also benchmarked a program to compute the greatest common denominator of two integers. This program required 25 cards, compiled without a listing in 5 seconds, and again executed in less than one second.

---

* John M. Gary, Department of Computer Science, University of Colorado.

A Dedicated Minicomputer System

We are now ready to consider the plausibility of supporting the lower division FORTRAN programming students on a minicomputer system. The primary consideration involved here are those of cost and efficiency. The proposed system should use an average budget of less than $15000/year if it is to be cost competetive with the existing central site computer system. (It is assumed that the system is used to support no other computer science course, where the total computer time budget is about $50000/year.) The system must provide good service to a minimum of 500 students per week, with growth possibilities to 750 students per week. Each student is expected to submit about 5 jobs per week (i.e. 63.3 jobs/semester), thus the minimum number of jobs per we is 2500, and the maximum is 3750 jobs per week. If these goals are met, the system can offer approximately the same service as the central site system. In addition, it is desirable that the dedicated system handle this load in a forty hour week, offering actual turnaround time approaching one minute. It is also desirable that each student be given the opportunity to enter his deck of cards into the card reader and burst his listing from the line printer. Operating in this mode, a dedicated system offers nearly instant batch turnaround to small groups of students using the machine in a given time period. Each student is also provided with a minimum degree of hands-on experience. It is this set of factors that weighs heavily toward a dedicated minicomputer system.

The Job Master File analysis indicates that a typical user program (during the course of the semester) is composed of about 30 cards and produces about 180 lines of output (including 40 lines of the banner page and accounting

information). The execution time of the program is insignificant. During peak demand periods near the end of the semester, the typical job may consist of about 36 cards and produce about 240 lines of output.

Taking the job load as a requirement on the system, we can estimate the required card reader rate, line printer rate, and processor throughput: Assuming that the system is in operation 40 hours per week, and that each student requires an average of 5 jobs per week, we obtain

$$\text{Number of jobs/week} = N_{week} = (\text{Number of students/week}) \times 5$$

$$\text{Number of jobs/minute} = N_{minute} = \frac{N_{week}}{2400}$$

$$\text{Card reader rate for semester} = CR_{ave} = 30\ N_{minute}$$

$$\text{Card reader rate for peak period} = CR_{max} = 36\ N_{minute}$$

$$\text{Line printer rate for the semester} = LP_{ave} = 140\ N_{minute}$$

$$\text{Line printer rate for peak period} = LP_{max} = 200\ N_{minute}$$

Here we ignore page eject times and the time required to remove an individual listing from the line printer. We also assume that a banner page and extensive accounting information is not printed.

To maximize resource overlap between the card reader, processor, and line printer operation, each job must be completely serviced by the processor (i.e. spooled in, scheduled, computed, loaded, executed, and spooled out) in

$$\frac{60}{N_{minute}} \text{ seconds.}$$

Table 3 summarizes these required service rates for various student loads considering average loads for the entire semester and peak loads for the end of the semester. The assumptions used here are:

-- The character of individual programs from semester-to-semester is static

-- No banner page nor extensive accounting log is to be printed

-- The card reader input rate is greater than or equal to the service rate of the system

-- Page eject time is negligible

During the fall semester of 1973, 513 students were enrolled in the lower division FORTRAN classes. From Table 3, it can be seen that a modest minicomputer system could handle the peak load, allowing about one minute of processor time for compilation, loading, etc. Furthermore, a very slow card reader is adequate, (it must read cards at a rate of about 40 cards/minute), and any faster card reader would create the need for substantial (infinite) disk space for job spooling. If the system were equipped with a single line printer which operated at a rate in the excess of 210 lines/minute, the printer could adequately handle the job load.

Note that the portion of the machine that must be drastically upgraded for increasing job load is the processor itself, provided that a reasonable card reader (say 75-100 cards per minute) and line printer (say 300 lines per minute) are initially configured into the machine. Recall that the major portion of processor time for the benchmark programs was employed in the loader; it is conceivable that as the current job load grows upward, advanced (graduate) students could be involved in improving both the compiler and loader efficiency in order to meet the processor requirement. Of course a better solution is to initially obtain a minicomputer with adequate software.

SUMMARY

In this paper we have discussed the job load imposed on a central site computer system by the lower division FORTRAN programming students. In particular, the number of jobs submitted for computer service was obtained, and the expected number of jobs per student was calculated. A typical job profile was derived, with respect to the number of cards in the job, the number of lines of output produced by the job, and an indicator of the processor requirement for three systems. Using this data, we have speculated a configuration of a minicomputer system that would handle the current job load as well as allowing for growth to a load that is 1.5 times the size of the current load. The main conclusion, then, is that a dedicated minicomputer can handle the growing load caused by lower division FORTRAN programming students, and that a careful choice of the system can provide an excess of processor resource which could be used for other purposes.

REFERENCES

1. Nutt, G. J., "The Boeing Dayfile Analysis Modified for the University of Washington CDC 6400", University of Washington Computer Center Technical Report No. 4, (Oct., 1970).

2. Nutt, G. J., "Computer System Monitoring Techniques", University of Colorado Department of Computer Science Technical Report No. CU-CS-013-73 (Feb. 1973).

3. Smith, M.E., "JOBLIST: Monthly Job Listing", DIGIT, University of Colorado Computing Center, Volume 8, No. 10, (June, 1973), page 7.

4. Watson, R. "Computer Performance Analysis: Application of Accounting Data", Rand Report No. R-573-NASA/PR (May, 1971).

Job Master File Format

(Some items of information that actually appear on a job description record are omitted here for brevity.)

1. Subaccount number
2. User number
3. Number of tapes assigned to the job
4. Priority of the job
5. Job name and identification
6. Batch or interactive job indicator
7. Day of the month
8. Number of card sectors read
9. Number of line sectors printed
10. Number of microfilm sectors printed
11. Number of card sectors punched
12. Central processor time (in tenths of seconds)
13. Mass storage sector reads/writes
14. Mag tape sector reads/writes
15. Time the job spent in the input/output queues and competing for a control point ("Internal turnaround time")
16. Average amount of memory allocated.
17. Accounting dollar charge.

Table 1

Program	CDC 6400				Minicomputer X			Minicomputer Y		
	Compile (CP)	Load & Execute (CP)	Compile (real)	Load & Execute (real)	Compile w/ Listing (real)	Load (real)	Execute (real)	Compile w/o listing (real)	Load (real)	Execute (real)
TEST1	0.077	0.156	1	1	6.0	13.2	1.8	9	35	2
TEST2	0.116	0.172	1	1	7.8	12.0	3.0	12	30	2
TEST3	0.237	0.319	2	1	13.2	12.6	7.2	18	37	4
TEST4	0.475	0.256	2	1	24.0	13.2	-	-	-	-

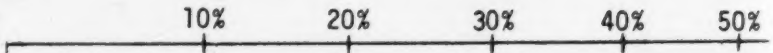
Test Program Timing Results

Table 2

BEST COPY AVAILABLE

BEST COPY AVAILABLE

Greater Than	Less Equal	Number	
	1	19	
1	2	0	
2	3	44	
3	4	2326	12.8 **********
4	5	3724	20.5 **********
5	6	4098	22.6 **********
6	7	3387	18.6 **********
7	8	2223	12.2 **********
8	9	1046	5.8 ******
9	10	618	3.4 ***
10	12	368	2.0 **
12	14	151	
14	16	54	
16	18	14	
18	20	6	
20	22	7	
22	24	5	
24	26	9	
26	-	65	



Mean: 184.8
Population: 18164

CARD COUNT IN SECTORS - SEMESTER

FIGURE 1

Students / week	Jobs / week (N_{week})	Jobs / minute (N_{minute})	Average card reader rate (CR_{ave}) cards/minute	Peak card reader rate (CR_{max}) cards/minute	Average line printer rate (LP_{ave}) lines/minute	Peak line printer rate (LP_{max}) lines/minute	Processor seconds per job
500	2500	1.04	31.2	37.4	145.6	208.0	57.7
550	2750	1.15	34.5	41.4	161.0	230.0	52.2
600	3000	1.25	37.5	45.0	175.0	250.0	48.0
650	3250	1.35	40.5	48.6	189.0	270.0	44.4
700	3500	1.46	43.8	52.6	204.4	292.0	41.1
750	3750	1.56	46.8	56.2	218.4	312.0	38.5

Required Service Rates

Table 3

BEST COPY AVAILABLE

BEST COPY AVAILABLE

Greater Than	Less Equal	Number	
	1	7	
1	2	0	
2	3	3	
3	4	77	2.1 **
4	5	83	2.3 **
5	6	360	10.0 **********
6	7	1300	36.2 **********
7	8	784	21.8 **********
8	9	411	11.5 **********
9	10	277	7.7 ********
10	12	182	5.1 *****
12	14	53	1.5 **
14	16	18	
16	18	6	
18	20	4	
20	22	2	
22	24	3	
24	26	5	
26		14	

10% 20% 30% 40% 50%

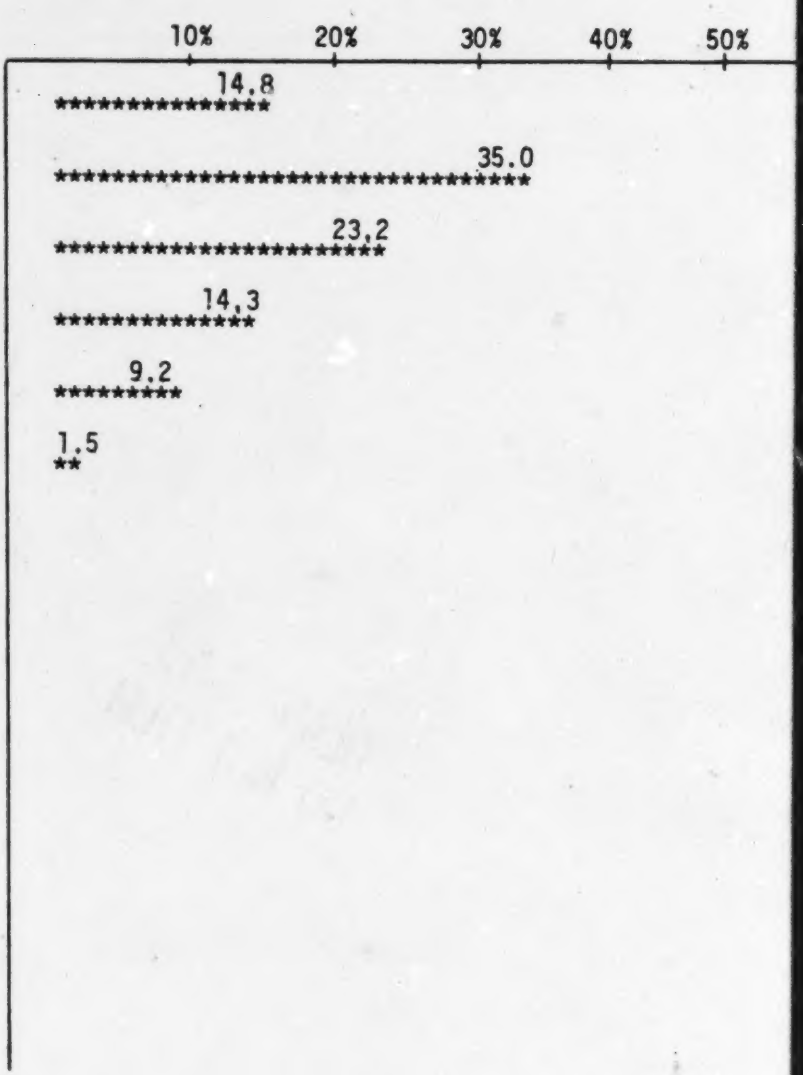
Mean: 69.29
Population: 3589

CARD COUNT IN SECTORS - MAY

Figure 2

BEST COPY AVAILABLE

Greater Than	Less Equal	Number
	5	2690
5	10	6350
10	15	4210
15	20	2603
20	30	1673
30	40	268
40	50	96
50	60	36
60	75	43
75	90	43
90	105	17
105	125	19
125	150	56
150	-	60



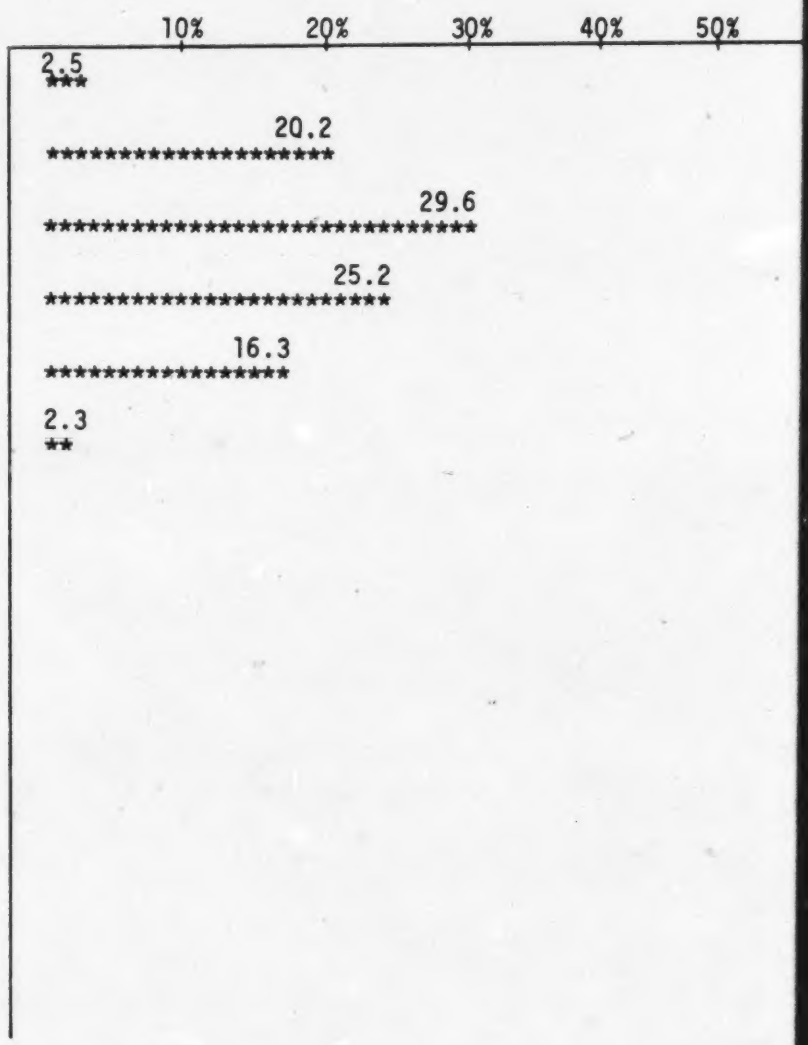
Mean: 13.88
Population: 18164

LINE COUNT IN SECTORS - SEMESTER

Figure 3

BEST COPY AVAILABLE

Greater Than	Less Equal	Number
	5	88
5	10	724
10	15	1063
15	20	905
20	30	584
30	40	81
40	50	33
50	60	15
60	75	19
75	90	19
90	105	4
105	125	9
125	150	26
150		20

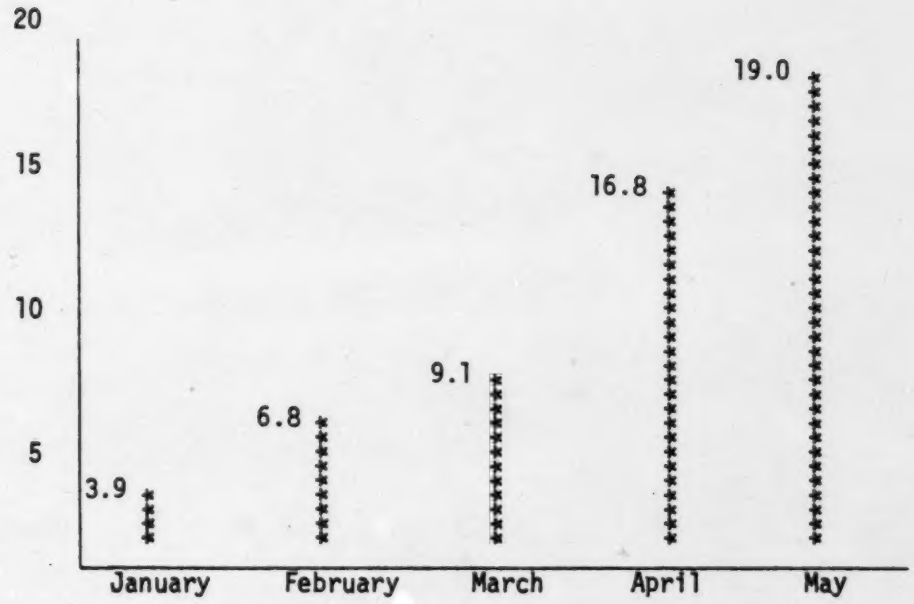


Mean: 19.01
Population: 3589

LINE COUNT IN SECTORS - MAY

Figure 4

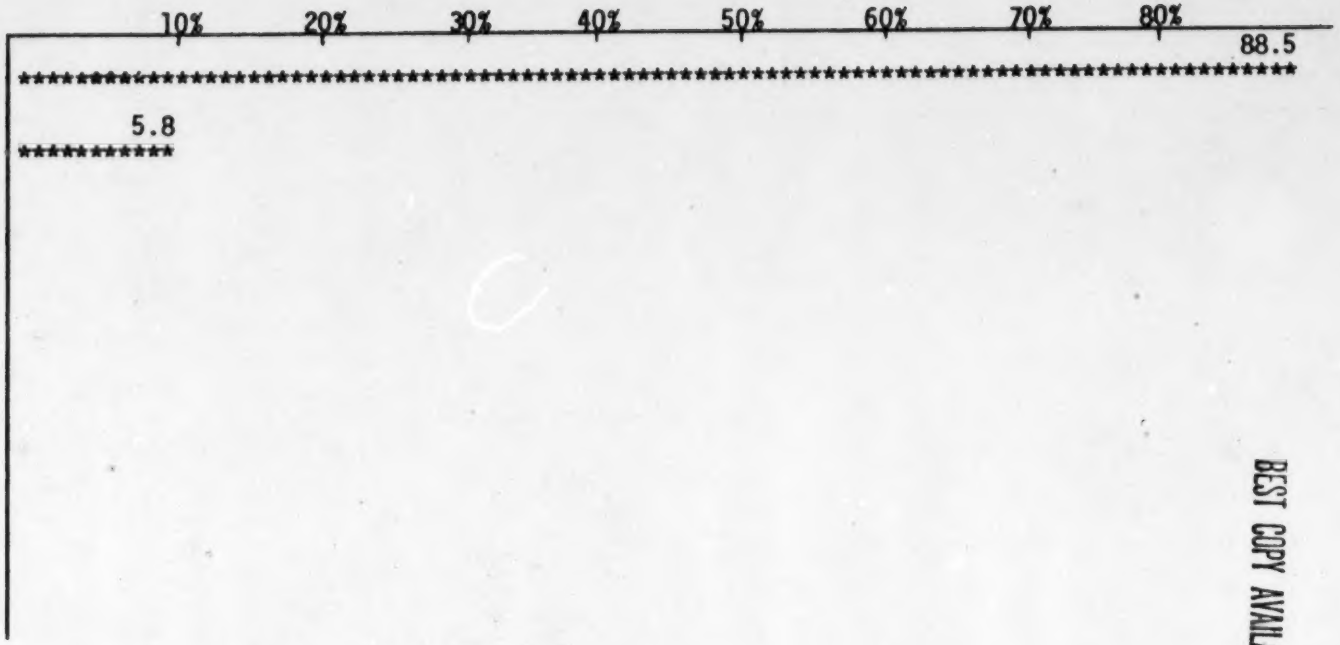
BEST COPY AVAILABLE



MEAN VALUE OF OUTPUT SECTORS /JOB BY MONTH

FIGURE 5

Greater Than	Less Equal	Number
	10	16083
10	20	1053
20	30	77
30	40	61
40	50	36
50	60	26
60	70	22
70	240	806



Mean: 9.77
Population: 18164

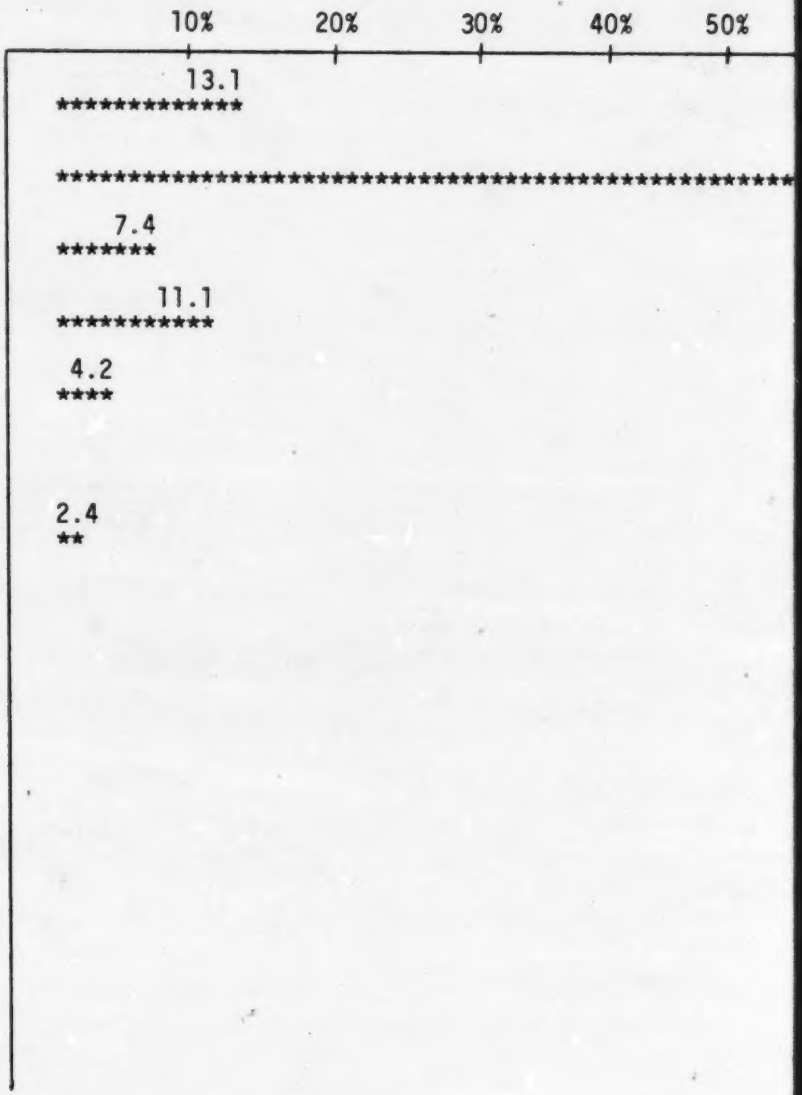
CP TIME IN HUNDREDTHS OF SECONDS - SEMESTER

FIGURE 6

BEST COPY AVAILABLE

BEST COPY AVAILABLE

Greater Than	Less Equal	Number	
	20	2372	13.1 **********
20	30	10618	**********
30	40	1345	7.4 *******
40	50	2010	11.1 **********
50	60	769	4.2 ****
60	70	67	
70	80	439	2.4 **
80	90	114	
90	100	54	
100	120	110	
120	140	78	
140	160	49	
160	180	36	
180		103	



Mean: 34.46
Population: 18164

ACCOUNTING CHARGE IN CENTS - SEMESTER

FIGURE 7